Threat modeling tools and approaches

Tools – first brief intro: commercial and open-source

- Commercial:
- IriusRisk – free edition, intuitive, efficient
- ThreatModeler – free edition, contrintuitive, complex components
- Totamantic – no free edition available to compare
- ThreAgile
- ThreatDragon
- Draw.io
- PyTM
- Microsoft TM tool
- PILLAR

Methodologies

- STRIDE
- ....
- PASTA
- LINDDUN

Approaches – how to introduce TM?

- RTM – Rapid threat modeling
- https://www.bluehatsecurity.net/blog/rapid-threat-model-prototyping
- Incremental TM
- Card games
- Classic approach (paper, STRIDE)

2 types of tools

- Threat modeling *from* code
- Threat modeling *with* code (aka threat modeling *in* code)

The results you get depend on the

- *quality of your input (the description of the system and its attributes) for the automation.*
- *algorithms and rules used in the analysis to be "correct," such that a given set of inputs generates valid and justifiable outputs.*

TM from code

- https://threatspec.org/ - 2019
- ThreatPlaybook https://we45.github.io/ThreatPlaybook/#/ – V2, 2020
- Not supported anymore

TM with code (model representation)

- Pytm https://github.com/OWASP/pytm
- Threagile https://threagile.io/
- Both continuously updated

Other tools: questionnaire, diagram

- Threatmodeler
- Microsoft TMT

- Commercial:
- IriusRisk + community edition
- SD elements
- Threatmodeler
- Tutamen
- Cairis

Continuous TM

- Autodesk CTM ???

Tool classification depending on task and time you have

MOST COMMON QUESTION HOW to do it in practice?

Demystify threat modeling

This will be our main criteria for comparison – can I do it in practice in small team?

How to make it feasible and continuous

- Threat modeling in practice requires seamless integration
- It is about the change in the mindset
- Let`s be practical

We have two teams

- One teams is agile and likes experiments
- They go threagile + incremental TM
- They improve the results of Threagile by using fast RTMP approach (permits integration).

- Other team has diagram, and answers the questions of customer/compliance/security -driven
- Fast analysis but in greater depth
- They go for IriusRisk
- They get detailed results but cannot use incremental approach or RTMP (questionas about complete app)

Main difference: risk appetite

Threagile includes approximately 40 built-in risk rules that analyze your architecture model to identify potential security risks. These rules cover a wide range of security concerns, such as:

DoS-risky Access Across Trust-Boundary

Missing Build Infrastructure

Cross-Site Scripting (XSS)

Unencrypted Communication

SQL/NoSQL-Injection

- For a complete list of these rules, you can refer to Threagile's risk rules documentation.
- Additionally, Threagile allows users to create custom risk rules to address specific security requirements unique to their projects.

- IriusRisk contains **thousands of built-in rules** that are integral to its threat modeling process. These rules automatically generate potential threats, assess risks, and suggest countermeasures based on your system's architecture and data flows.
- Additionally, IriusRisk offers a robust interface for developing custom rules, allowing you to tailor the threat modeling to your organization's specific needs.

Criteria	Threagile + RTMP + Incremental	IriusRisk	Complete TM Scheme (DFD + STRIDE)
Setup Time	⭐ Quick YAML + docs + session	🕓 Medium UI config + rules engine setup	⏳ Long Draw full system upfront
Focus	🎯 Feature-first + evolving system (Incremental, Agile-friendly)	🧠 System-driven with rules-based coverage	🗺️ Top-down architecture
Input Method	YAML + structured RTMP inputs + team sessions	UI + questionnaires + asset library	Diagrams (DFD) + security annotations
Coverage	✅ Custom risks, abuse cases, STRIDE + OWASP ASVS	✅ Rule-based auto threats, regulatory mapping	✅ Exhaustive system-level threats
Adaptability	🔄 Easy to evolve incrementally	⚙️ Templated but modifiable	❌ Hard to maintain with system changes
Automation & CI/CD	✅ Very scriptable (Threagile CLI + Git)	✅ Integrates with Jira, CI/CD	❌ Typically manual
Collaboration	🤝 Dev-first + threat modeling workshops	🧩 Security engineer-led with workflow roles	🧱 Often isolated security team task
Detail Level	⚖️ Balanced – custom + templated	🎯 High-level threats, lower granularity unless fine-tuned	📚 Very detailed but can become overwhelming
Learning Curve	🟢 Low (Threagile + RTMP are beginner-friendly)	🟡 Medium (need to learn rule model and UI)	🔴 High (requires security background)
Documentation Output	📄 Full risk reports + diagrams + YAML versioning	📊 Dashboard + exportable risk lists	📑 Threat tables + DFDs
Cost	🆓 Free (open source)	💰 Freemium → paid plans	❌ Cost is time + skilled resources
Best Use Case	Agile teams, evolving codebase, collaborative TM	Compliance, repeatable enterprise TM	Formal risk review, regulated systems

Example 1

🧱 The Stack

Let's say your team is building a **basic to-do app**:

- **Frontend**: React app in the browser
- **Backend**: Node.js API server
- **Database**: PostgreSQL

Incremental TM example: Sharing to-do list feature

✅ Step 1: Define the Feature

- As a user, I want to share my to-do list with a friend via email so they can view it online.

🔍 Step 2: Analyze the Feature Incrementally

In ITM, you **only analyze the security risks of *this feature* and its impact on the system**.

New Components / Changes:

- New API endpoint: POST /share
- Backend emails a unique link with access token
- New DB table for tracking shared links and expiration

🔎 Step 3: Apply a Lightweight STRIDE Pass

🔐 Step 4: Identify Risk & Mitigations

📌 Step 5: Capture It

Write down the model in:

- A shared Notion doc or markdown file in Git
- Or even inline in the GitHub issue for the feature!

Recap – STRIDE and risk mitigations

STRIDE Category	Potential Threat	Example
Spoofing	A user fakes another user's identity when sharing	Is auth checked on `/share`?
Tampering	Token tampered with in URL	Is it signed or verified server-side?
Repudiation	User denies they shared the list	Do we log who created the link and when?
Information Disclosure	Shared list can be accessed by anyone with the link	Is there a token expiration? One-time use?
Denial of Service	Shared links spammed or misused	Are there rate limits on sharing?
Elevation of Privilege	Shared user edits the list, even if only view access	Is access control enforced properly?

Risk	Mitigation
Link guessing or token leakage	Use long, unguessable tokens; store tokens securely; HTTPS only
No expiration	Add expiry date to links
No logging	Add audit logs for share actions
Too much access	Backend checks "read-only" scope on token

Why it is good? – Organic coverage!

- Sprint by sprint, you've layered in just-enough threat modeling.
- No giant up-front effort.
- Your system-level model grows *alongside* your architecture.

Capture this model in yaml file (Threagile)

A new API endpoint (/share)

A database entry for sharing metadata

Email delivery of a tokenized link

+ ChatGPT to generate snippet

🧩 **Threagile Snippet: Incremental Model Update**

yaml Копировать

```yaml
title: To-Do App Threat Model
author: Small Dev Team
date: 2025-03-24

# New technical assets
technical_assets:
  ShareAPI:
    type: process
    trust_zone: BackendZone
    usage: business
    data_assets_processed: [ToDoData, ShareMetadata]
    technology: web-service-rest
    internet: false

  MailerService:
    type: process
    trust_zone: BackendZone
    usage: business
    data_assets_processed: [EmailContent]
    technology: external-service
    internet: true
```

Several rounds of refinement to correspond to schema

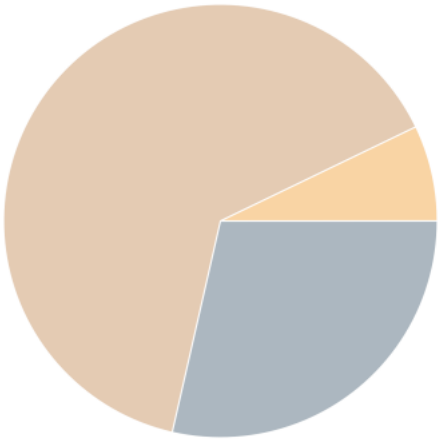
- https://run.threagile.io/

Risks by Vulnerability Category

Identified Risks by Vulnerability Category

- Cross-Site Scripting (XSS): 1 / 1 Risk
- Container Base Image Backdooring: 1 / 1 Risk
- Missing Build Infrastructure: 1 / 1 Risk
- Missing Cloud Hardening: 3 / 3 Risks
- Missing Hardening: 2 / 2 Risks
- Missing Vault (Secret Storage): 1 / 1 Risk
- Unencrypted Technical Assets: 1 / 1 Risk
- Unnecessary Technical Asset: 4 / 4 Risks

0 critical risk
0 high risk
1 elevated risk
9 medium risk
4 low risk



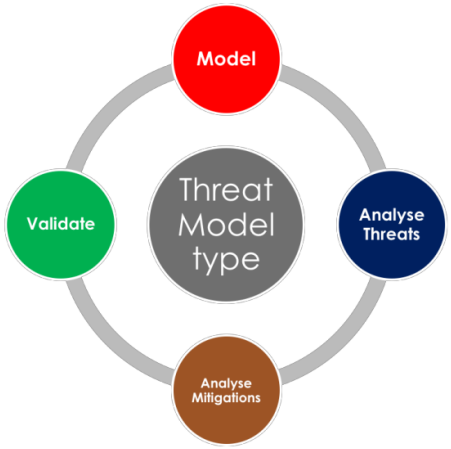
- data-asset-diagram.png
- data-flow-diagram.png
- report.pdf
- risks.json
- risks.xlsx
- stats.json
- tags.xlsx
- technical-assets.json
- threagile.yaml

What lacks here?

- Risks are purely technical (no business side)
- We can improve the analysis by adding to the model what is important for us

Rapid TM prototyping

- Rapid Threat Model Prototyping (RTMP) is a streamlined approach to threat modeling that emphasizes efficiency and integration within Agile and DevOps workflows.
- https://www.bluehatsecurity.net/blog/rapid-threat-model-prototyping



Step 1 (flows, sources, sinks and zones)

- basic flows (showing the direction of execution) and processes
- the sources (where the flow starts)
- the sinks (where the data is stored/ done processing it)

- zone 0: elements outside of control (external/ anonymous users, other systems)
- zone 1: boundary zones between trusted and untrusted (such as DMZ elements)
- zone > 1: the more critical the systems, the higher the assigned zone

assign to sources the lowest zone of trust, to sinks the highest zone of trust and to flows the origin zone of trust.

Step 2: STRIDE + rules

- elevation of privilege rule – start with this one; where there is a positive difference between the destination zone of trust and the source one, place “E” on the destination process (the flow is towards a more critical system so authorization is needed in place)
- spoofing rule: place “S” on the destination process where the origin has zone of trust 0 (the origin is a zone outside of control so authentication is needed)
- tampering rule: place “T” on connnecting flows where there is a positive difference between the destination zone of trust and the source one (data can be tampered in transit from source to destination, integrity is needed)
- repudiation rule: place “R” on destination where there is a “T” on the flow and a “S” on the destination (data is tampered in transit and repudiated afterwards, integrity and authentication is needed)
- information disclosure: place “I” on destination where there is a negative difference between the destination zone of trust and the source one (data from a more sensitive system is sent to a less sensitive system, confidentiality is needed) denial of service rule: place “D” on destination where the source is zone 0 (no control over the source actions, availability is needed)

Step 3: Analyze vulnerabilities using OWASP Top10

OWASP TOP 10	STRIDE
A1 - Injection	1. Tampering 2. Elevation of Privilege 3. Denial of Service
A2 - Broken Authentication	1. Spoofing 2. Repudiation 3. Information Disclosure
A3 - Sensitive Data Exposure	1. Information Disclosure 2. Spoofing 3. Elevation of Privilege
A4 - XML External Entities (XXE)	1. Tampering 2. Elevation of Privilege 3. Information Disclosure
A5 - Broken Access Control	1. Elevation of Privilege 2. Repudiation 3. Tampering
A6 - Security Misconfiguration	1. Elevation of Privilege 2. Spoofing 3. Information Disclosure
A7 - Cross-Site Scripting (XSS)	1. Tampering 2. Elevation of Privilege 3. Repudiation
A8 - Insecure Deserialization	1. Tampering 2. Spoofing 3. Elevation of Privilege
A9 - Using Components with Known Vulnerabilities	1. Elevation of Privilege 2. Spoofing 3. Denial of Service
A10 - Insufficient Logging & Monitoring	1. Repudiation 2. Information Disclosure 3. Tampering

+ Step 4 – Validate what applies

Back to our example

🧱 1. System Overview

Architecture Tier Involved:

- **Frontend**: React app
- **Backend**: Node.js REST API (/share endpoint)
- **Database**: PostgreSQL storing sharing metadata
- **Mailer Service**: Sends a tokenized link via email

Zones of trust

Zone	Components
Zone 0 – Untrusted	Public Internet, external users
Zone 1 – Web Tier	React Frontend
Zone 2 – App Tier	Backend API (`/share` endpoint), Mailer
Zone 3 – Data Tier	PostgreSQL

STRIDE analysis

Category	Threat	Component	Description
S – Spoofing	Fake user triggers share	`/share` endpoint	Weak auth may allow impersonation
T – Tampering	Token modified in URL	Link in email	Malicious user may try to elevate access
R – Repudiation	User denies sharing	Backend API	No logging of share actions
I – Info Disclosure	Token leaks or is reused	Email/Token	Anyone with link gains access
D – Denial of Service	API spammed with share requests	Backend	Could exhaust rate limits or emails
E – Elevation of Privilege	Viewer gains edit rights	API	Link may not properly restrict scope

Risk register

Risk ID	Description	Likelihood	Impact	Risk Level	Recommended Mitigations
R1	Token in URL is predictable or reused	Medium	High	**High**	Use UUIDv4 or signed JWT; one-time use; short expiration
R2	No authentication or access control checks on `/share`	Low	High	**Medium**	Enforce user authentication and permissions
R3	No audit trail of who shared what and when	Medium	Medium	**Medium**	Log IP, user ID, timestamp, link created
R4	Link is leaked in referer headers or logs	Medium	High	**High**	Avoid putting token in URL path; use POST + auth or short-lived query param
R5	Shared user can modify content	Medium	High	**High**	Enforce read-only permission on shared links
R6	Mass sharing floods email service	Low	Medium	**Low**	Rate limit `/share`; CAPTCHA or auth token before sending

Mitigations

Area	Suggested Controls
Token Security	Long random tokens (UUIDv4 or signed), expiration, one-time use
Access Control	Backend validates user identity; enforce read-only permissions
Audit & Logging	Log share action: user ID, email target, token generated
Rate Limiting	Prevent abuse via brute-force or bulk requests
Token Handling	Avoid leaking tokens in URLs (use POST or Authorization headers)
User Feedback	UI should notify user that shared links may expose data

Why?

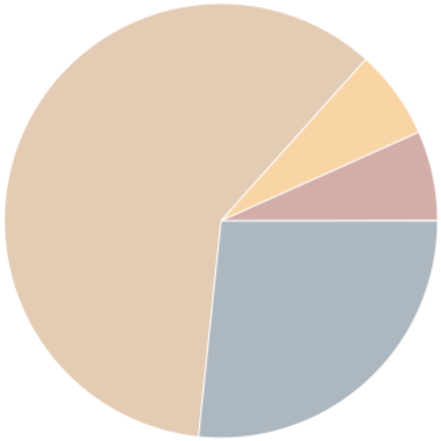
- Quick
- Human-oriented
- Feature-specific risks
- Finds other risks than standard Threagile

- **embed RTMP outputs into the Threagile model** as custom risks, abuse cases, or security requirements.

Resulting report

Risks by Vulnerability Category

0 critical risk
1 high risk
1 elevated risk
9 medium risk
4 low risk

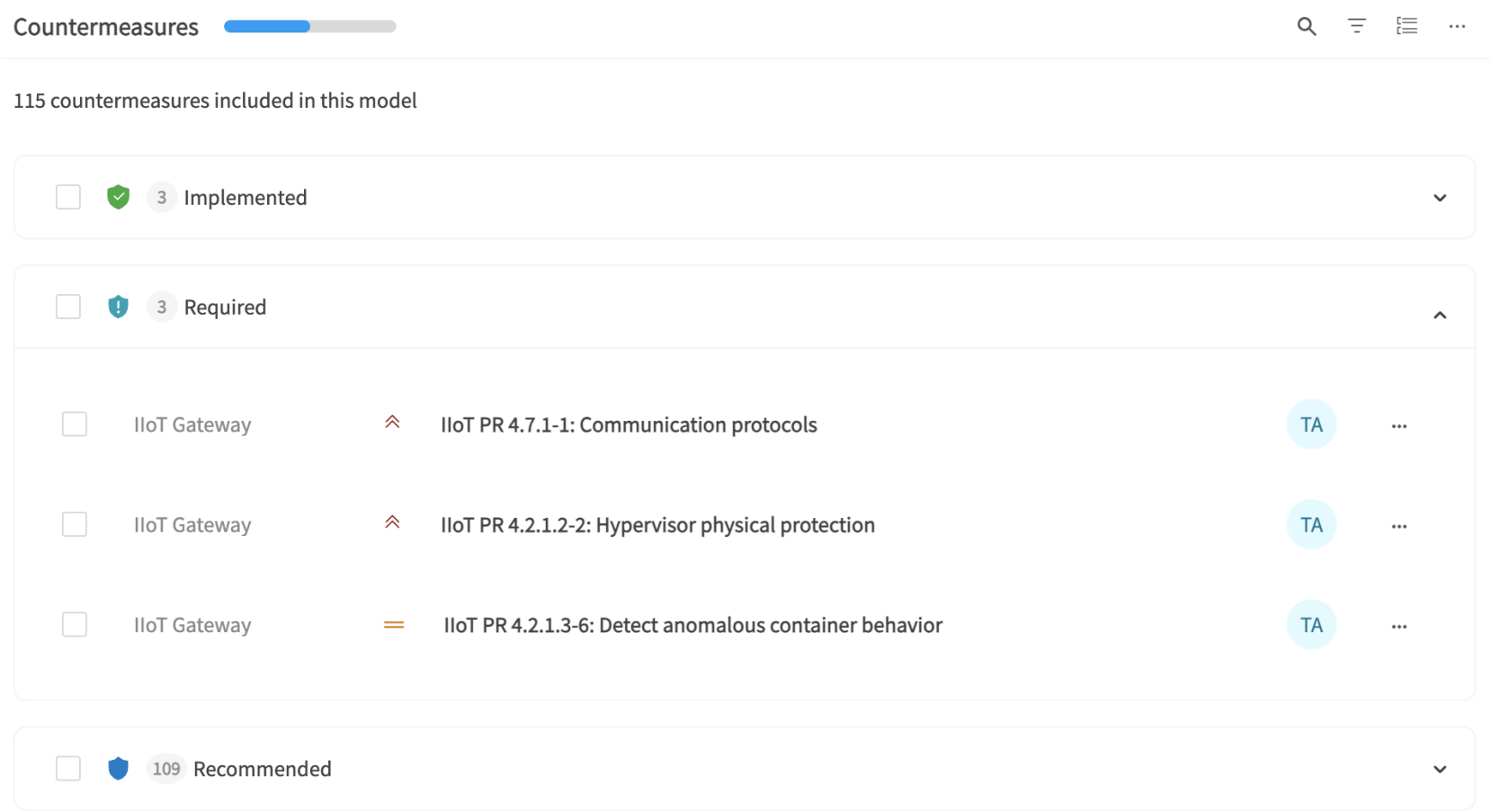


Example IoT app – Irius risk

Draw a scheme

--Import template or drag an drop
--trust zones - very important
--answer questions per component
--update model
--risks are identified and prioritized:

Risks


Countermeasures
115 countermeasures included in this model
3 Implemented
3 Required
IIoT Gateway
IIoT PR 4.7.1-1: Communication protocols
TA
IIoT Gateway
IIoT PR 4.2.1.2-2: Hypervisor physical protection
TA
IIoT Gateway
IIoT PR 4.2.1.3-6: Detect anomalous container behavior
TA
109 Recommended

Threat details

Risk exposed | Save

IIoT Gateway / Information Disclosure

Eavesdropping

Critical risk

Description

Cleartext transmission of sensitive information can result in data interception by attackers

Countermeasures

0%

1 CWE-319

IIoT Gateway		IIoT PR 4.7.1-1: Communication protocols	Required	100%	TA	

Comments | References | History

TA Add comment

Expiry date: Select date

STRIDE-LM: Information Disclosure

MITRE reference: ATT&CK Enterprise - T1052 - Exfiltration Over Physical Medium

Impact

* Confidentiality: Very high
* Integrity: Very high
* Availability: Low
* Ease of Exploitation: Very high

Choose most important actions

- IoT gateway
- Sensor
- SQL database
- External parties
- WHAT to do, how much time will it take

Development teams

- Challenges¶
- Threat modeling can be challenging for development teams for several key reasons. Firstly, many developers lack sufficient knowledge and experience in the field of security, which hinders their ability to effectively use methodologies and frameworks, identify, and model threats. Without proper training and understanding of basic security principles, developers may overlook potential threats or incorrectly assess their risks.
- Additionally, the threat modeling process can be complex and time-consuming. It requires a systematic approach and in-depth analysis, which is often difficult to reconcile with tight schedules and the pressure to deliver new functionalities. Development teams may feel a lack of tools and resources to support them in this task, leading to frustration and discouragement.
- Another challenge is the communication and collaboration between different departments within the organization. Without effective communication between development teams, security teams, and other stakeholders, threat modeling can be incomplete or misdirected.

How to

- inviting members of the security teams to threat modeling sessions, build a culture of collaboration and mutual support within the organization, leading to a more comprehensive approach to security.
- regular IT security training for their development teams tailored to the specific needs of the team.
- implement processes and tools that simplify and automate threat modeling.
- It is also important to promote a culture of security throughout the organization, where threat modeling is seen as an integral part of the Software Development Life Cycle (SDLC), rather than an additional burden.
- Regular review sessions and cross-team workshops

In practice

- Workshops – how to make an in house workshop
- Tools – which tools to use and how
- Knowledge/culture – other tips – communication – collaboration, regular programme, reports should be shareable

How to do it fast and in less time?

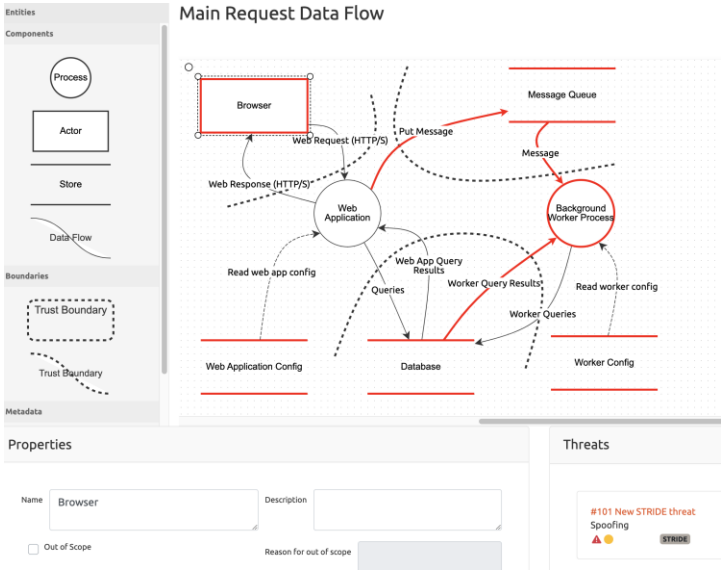
- Typically threat modeling is blablabla

- Ways to do it faster:
- ---automate
- ---simplify (rapid threat modeling+ security prompts)

- --- Incremental threat modeling and security prompts
- https://2017.appsec.eu/presos/CISO/Incremental%20Threat%20Modelling%20-%20Irene%20Michlin%20-%20OWASP_AppSec-Eu_2017.pdf
- https://www.linkedin.com/pulse/irene-mitchin-navigating-complexities-threat-ai-age-cipollone-z0xof/
-
- Security prompts in agile development - podcast
- Rapid threat modeling prototyping
-
- https://github.com/geoffrey-hill-tutamantic/rapid-threat-model-prototyping-docs

Links threat modelling

- https://cheatsheetseries.owasp.org/cheatsheets/Threat_Modeling_Cheat_Sheet.html
- https://owasp.org/www-project-threat-model-cookbook/
- https://owasp.org/www-project-threat-dragon/
- https://www.youtube.com/watch?v=ARjlRFQN7XM



A curated list to dig in

- https://github.com/hysnsec/awesome-threat-modelling?tab=readme-ov-file
- A book : https://learning.oreilly.com/library/view/threat-modeling/9781492056546/foreword01.html
- A workshop AWS: https://explore.skillbuilder.aws/learn/courses/13274/threat-modeling-the-right-way-for-builders-workshop

Comments

- Overview of tools or a way to use them?
- Focus more on the tools for TM
- Later – do a session on 2 practical approaches
- We can even have a workshop later in zweenarde
- …